Remarks

Status of application

Claims 1-99 were examined and stand rejected in view of the prior art. The claims were previously amended to further clarify Applicant's invention. Reexamination and reconsideration are respectfully requested.

The invention

For a brief summary of Applicant's invention, please refer to the Amendment filed on September 15, 2008.

Prior art rejections

A. First Section 103 rejection: Newman in view of Lei

Claims 37-70 are rejected under 35 U.S.C. 103(a) as being anticipated by Newman et al. (US Patent 7,266,699 B2, "Newman") in view of Lei et al. (US Publication 2004/0255133 A1, "Lei"). Here, the Examiner reasserts the previous rejection (of claims 1-99) based on the combination of Newman and Lei (i.e., combining Newman's method of providing a "transparent" encryption infrastructure for databases with Lei's method of storing and updating encrypted tables). For the reasons set forth below, the claims of this group distinguish over the combined references.

Applicant appreciates that the Examiner's job is to accord Applicant's claim language broad (or exceedingly broad) scope. However, upon reviewing the pending claims and the cited prior art, it is respectfully submitted that the Examiner has given far too much credit to the prior art than is justified based on a careful analysis of the art references themselves. This discrepancy will now be discussed.

In Applicant's last-filed Amendment, Applicant amended all independent claims to clarify that Applicant's claimed invention is directed to **SQL extensions** that support creation and management of **named** encryption keys, thereby providing improved column-based encryption. Here, the term "named" key, as described in Applicant's specification and used in Applicant's claims, means that the key can actually be referenced in a syntactically correct manner within SQL statements provided by users. Very specific use of named keys in SQL statements is provided in Applicant's dependent

claims, for example:

9. (Original) The method of claim 8, wherein the CREATE TABLE command includes:

CREATE TABLE tablename
(colname1 datatype **[encrypt [with [db.[owner].]keyname]**,
colname2 datatype **[encrypt [with [db.[owner].]keyname]**)
as its syntax.

(Emphasis added.)

As shown above, the named key (*keyname*) is a syntactic construct used within an SQL CREATE TABLE statement. This is very different from anything described by Newman or Lei.

Why does a named key approach even matter? There are several important practical implications to the approach. For example, apart from the SQL statement used to create the named key (Applicant's CREATE ENCRYPTION KEY statement -- the subject matter of several claims), the named encryption key that has been created may be referenced (i.e., used) in other SQL statements, such as CREATE TABLE and ALTER TABLE statements (the subject matter in several claims, including for example 8, 9, 11, 12, 44, 45, 47, 48, 83 and 84), as well as other key-specific statements such as GRANT ALTER ENCRYPTION KEY. Here, other SQL statements that are performing a database task (e.g., creating tables) may reference a particular named key as the encryption key used for column encryption of various columns that are to be encrypted. Even the creation and management of the encryption key itself are achieved through SQL statements. This approach greatly simplifies the creation and management of encryption keys, since the database user can perform **all** encryption-related tasks through familiar SQL statements without requiring any new or special client software.

Detailed review of Newman indicates that his solution is very similar to the Oracle DBMS_OBFUSCATION_TOOLKIT described in Applicant's Background Section. In fact, Newman states that his invention "is similar in concept to the

DBMS_OBFUSCATION_TOOLKIT." (Newman, column 1, line 60.) The differences between Newman's invention and the DBMS_OBFUSCATION_TOOLKIT are cataloged by Newman at column 1, line 62 to column 2, line 16. None of these differences or improvements provided by Newman pertains to SQL extensions or named encryption keys. Significantly, Newman discloses at column 4, lines 37-43 that his approach requires a separate client front end piece, in addition to a server module. This makes it very clear that Newman's approach does not provide anything akin to Applicant's SQL-based approach that is completely self-contained in the database server (no special client piece required), thus allowing the database user to perform all steps pertaining to the creation, management, and use of encryption keys via SQL statements that the user is accustomed to using (and without any additional or special client software). Should the Examiner have any doubt regarding Applicant's contention that Newman does not provide SQL extensions for creating, managing, and using named encryption keys, it is respectfully requested that the Examiner review Newman at column 4, lines 49-56, where Newman describes in detail that the Newman user employs the special client front end to create an encryption key (and one which has no "name" or other syntactically-significant identifier attached to it that would allow it to be used directly within SQL statements). Any contention that Newman somehow provides SQL extensions directly supporting the creation, management, and use of named encryption keys (as required by Applicant's claims) flies in the face of what Newman himself describes. Simply put, Newman does not teach or suggest SQL extensions (of any type) nor does he teach or suggest a named encryption key that can be used within SQL statements, as required by Applicant's claims.

The Examiner turns to Lei for the proposition that it teaches "for creating one or more database tables having particular column data encrypted with said particular named column encryption keys." As best as understood, the Examiner's citation of Lei is not for the foregoing deficiencies pertaining to (Applicant's claimed features of) SQL extensions supporting creation, management, and use of named encryption keys, as the Examiner contends that those are already present in Newman (Newman's disclosure notwithstanding). In any event, a review of Lei shows that that reference does nothing to cure these deficiencies of Newman, as now will be described.

Lei describes the use of ENCRYPT and DECRYPT commands in SQL statements (e.g., an SQL CREATE TABLE statement). Importantly, however, Lei has no notion of a named encryption key, or any other construct that would function in the same manner. Lei's SQL examples (including the one cited by the Examiner at paragraph [0066]) make this very clear. In each instance, the SQL statement containing either the ENCRYPT or DECRYPT command does not reference any sort of named key or any key. In fact, Lei shows at paragraph [0061] that his system internally hides this detail from the user. For example, the ALTER TABLE SQL command does not contain any sort of named key (surfaced at the level of the SQL statement) but instead internally his system translates that command into an UPDATE statement that retrieves a master key using an internal key retrieval operator (i.e., internal GET_KEY function). From the description and examples provided, it is clear that Lei does not support SQL-based creation or management of any encryption keys (named or otherwise), as required by Applicant's claims. Lei does describe SQL-based ENCRYPT and DECRYPT commands, but such commands never surface the encryption key in the SQL statement itself. Instead, Lei assumes these keys are created and managed elsewhere (i.e., outside of SQL statements).

All told, the particular failing of prior art systems, such as Newman and Lei, is in the creation and management of column encryption keys. Those systems have given a lot of thought and effort to integrating encrypting and decrypting operations, but have given practically no thought to the management of the encryption keys themselves. They contain absolutely no description whatsoever pertaining to use of SQL extensions to create encryption keys (contrast with Applicant's CREATE ENCRYPTION KEY native SQL command) or manage such encryption keys (contrast with Applicant's ALTER ENCRYPTION KEY native command). Instead, those systems treat encryption keys as more or less an afterthought and assume that they can be obtained from some external source (e.g., the "master keys" Lei assumes are available). Those prior art systems lose an important opportunity to improve the efficiency and operation of database encryption.

Under Section 103, the combined references must (among other things) teach or suggest all elements present in Applicant's claims. For the reasons stated, it is respectfully submitted that Applicant's named encryption key approach -- where a named encryption key is created, maintained, and used directly within SQL statements -- is not

shown in the prior art and thus represents a patentable advance in the area of database encryption and security. Accordingly, it is believed that the claims are patentable under Section 103.

B. Second Section 103 rejection: Newman in view of Sato and Lei

Claims 1-36 and 71-99 stand rejected under 35 U.S.C. 103(a) as being anticipated by Newman et al. (above) in view of Sato et al. (US Patent 7,093,137 B1, "Sato") and further in view of Lei et al. (above). Here, the Examiner repeats the rejection based on the combination of Newman and Lei, but adds Sato for the proposition that it teaches the claim limitation pertaining to the named encryption key capable of encrypting multiple columns. The claims are believed to be allowable for at least the reasons stated above pertaining to the first Section 103 rejection based on Newman and Lei. In particular, the combination of Newman and Lei fails to teach Applicant's approach of using SQL extensions that allow all aspects of creating, maintaining, and using encryption keys to be performed entirely within the confines of SQL statements written by database users. Sato contains no disclosure that remedies these deficiencies.

Applicant's claims are also believed to be allowable for the following additional reasons. A review of Sato shows that it does not teach the notion of using one key to encrypt multiple columns (besides of course lacking any teaching pertaining to a named encryption key). In particular, Sato discusses encryption using specific row keys. When Sato discusses encrypting a frequently-retrieved column with a "common key," he is discussing the notion of foregoing the "specific row keys" for this frequently-retrieved column. In other words, a single or common key is used to encrypt/decrypt all of the row items of a given column. He is not talking about a "common key" shared among several columns. Thus, based on the actual disclosure that Sato provides, the Examiner's contention that Sato teaches Applicant's claim limitation pertaining to using a named encryption key for encrypting multiple columns simply finds no support. Accordingly, it is respectfully submitted that the currently-rejected claims of this group distinguish over the combination of Newman, Sato, and Lei, and are thus patentable under Section 103.

Any dependent claims not explicitly discussed are believed to be allowable by

virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and prior amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: March 9, 2009

/John A. Smart/

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX